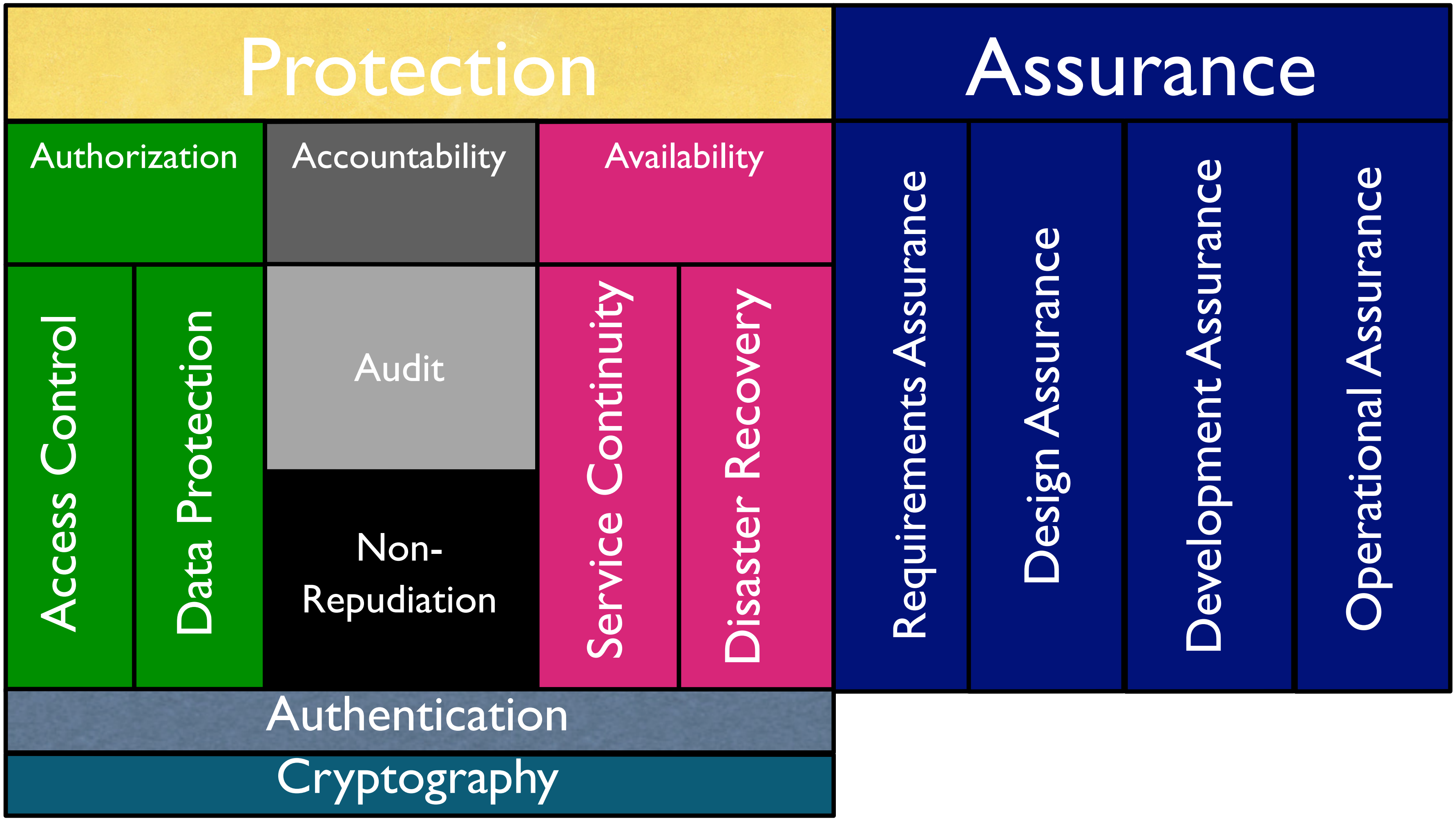
Authenticating People

EECE 412 “Introduction to Computer Security”

Konstantin Beznosov

where we are


Protection
Assurance
Authorization
Accountability
Availability
Access Control
Data Protection
Audit
Non-Repudiation
Service Continuity
Disaster Recovery
Requirements Assurance
Design Assurance
Development Assurance
Operational Assurance
Authentication
Cryptography

Basics and Terminology

definition

authentication is binding of identity to subject

- Identity is that of external entity
- Subject is computer entity
- Subject a.k.a. principal

What Authentication Factors are used?

- What you know
- What you have
- What you are

one-time passwords

- $h^1(m) = h(m)$
- $h^2(m) = h(h^1(m)) = h(h(m))$
- …
- $h^n(m) = h(h^{n-1}(m))=h(h(h^{n-2}(m)))$ …

http://upload.wikimedia.org/wikipedia/commons/8/8a/RSA_SecurID_Token_Old.jpg

- $p_1 = h^n(m)$, $p_2 = h^{n-1}(m)$, … $p_n = h^1(m)$

what you are (biometrics)

- Android liveliness check
  - https://www.youtube.com/watch?v=zYxphDK6s3I
- iPhone 5s TouchID
  - https://www.youtube.com/watch?v=baio0qUj2Lk

Password-based Authentication

What's Password?

- Lots of things act as passwords!
  - PIN
  - Social security number
  - Mother's maiden name
  - Date of birth
  - Name of your pet, etc.

- **Sequence of words**
- Examples: pass-phrases
- **Algorithms**
- Examples: challenge-resp

illustration: and now something completely different

- Monty Python and the Holy Grail (1h18m)

Why Passwords?

- Why is “something you know” more popular than “something you have” and “something you are”?
- **Cost**: passwords are free
- **Convenience**: easier for SA to reset password than to issue new smartcard

adversary model

- objectives
  - compromise any account(s) on a system
  - compromise specific account
- capabilities
  - before the attack
    - password cracking tool(s)
    - access to previously leaked/compromised passwords
  - during the attack
    - password cracking tool(s)
    - ability to perform off-line dictionary attacks on the password database, if leaked/compromised
    - ability to perform online dictionary attacks
    - knowledge of account names

Attacks on Passwords

- Attacker could…
  - Target one particular account
  - Target any account on system
  - Target any account on any system
  - Attempt denial of service (DoS) attack
- Common attack path
  - Outsider → normal user → administrator
  - May only require **one** weak password!

off-line cracking attacks on password databases

Keys vs Passwords

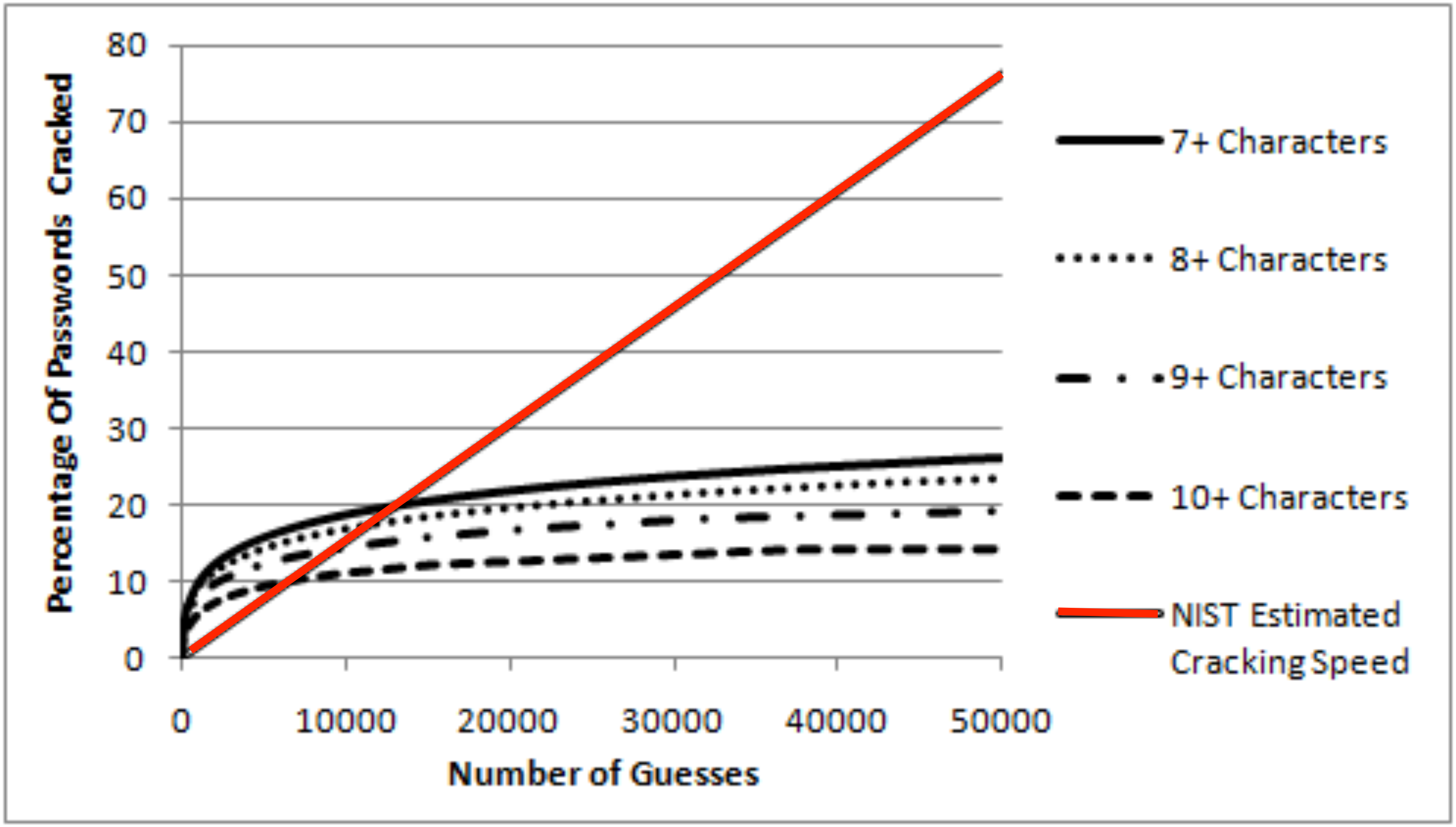
Crypto keys

- Suppose key is 64 bits
- Then 2^{64} keys
- Choose key at random
- Then attacker must try about 2^{63} keys

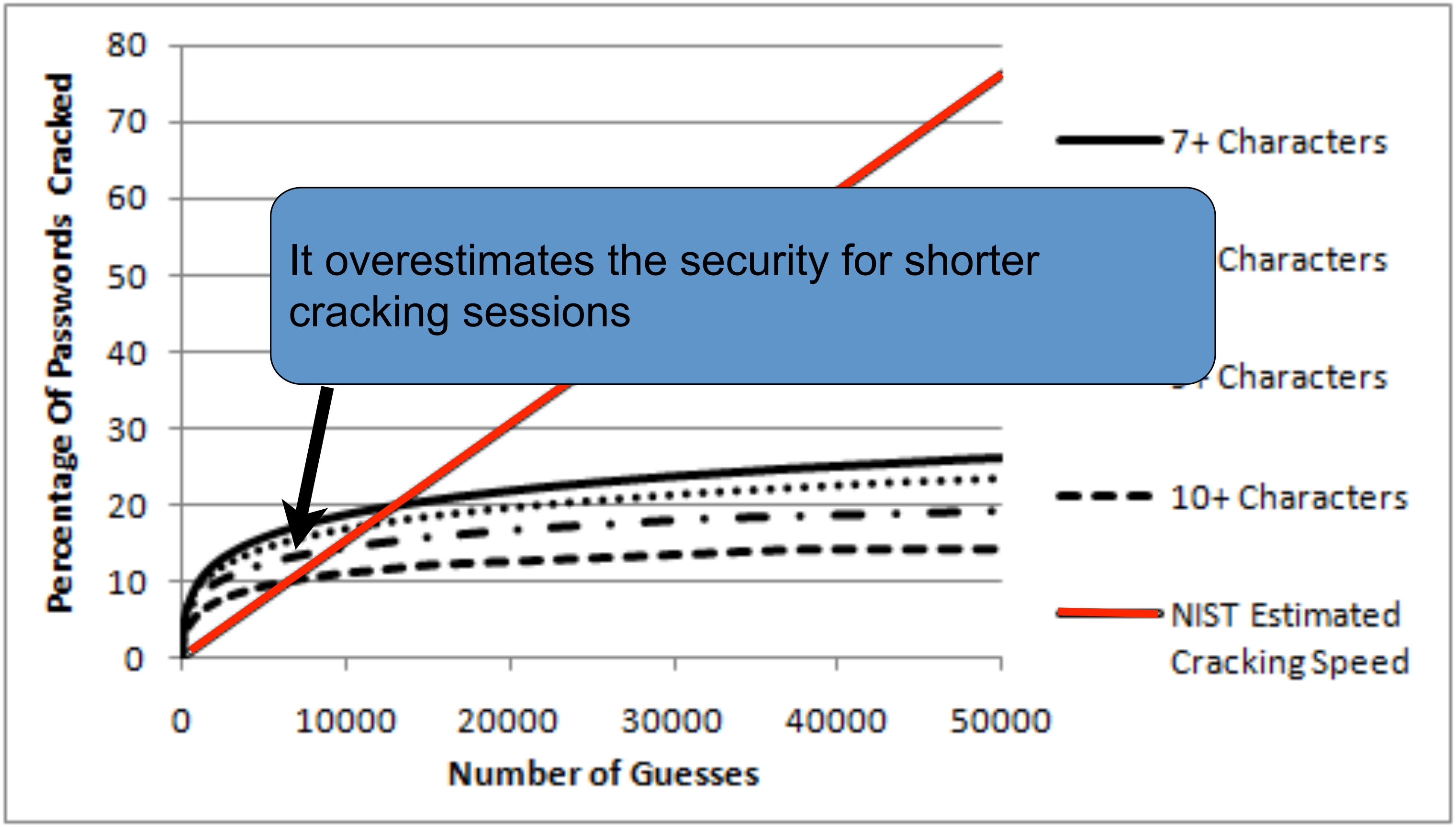
Passwords

- Suppose passwords are 8 characters, and 256 different characters
- Entropy is $\log_2(b^n)$
- Then $256^8 = 2^{64}$ pwds

Where this Breaks Down

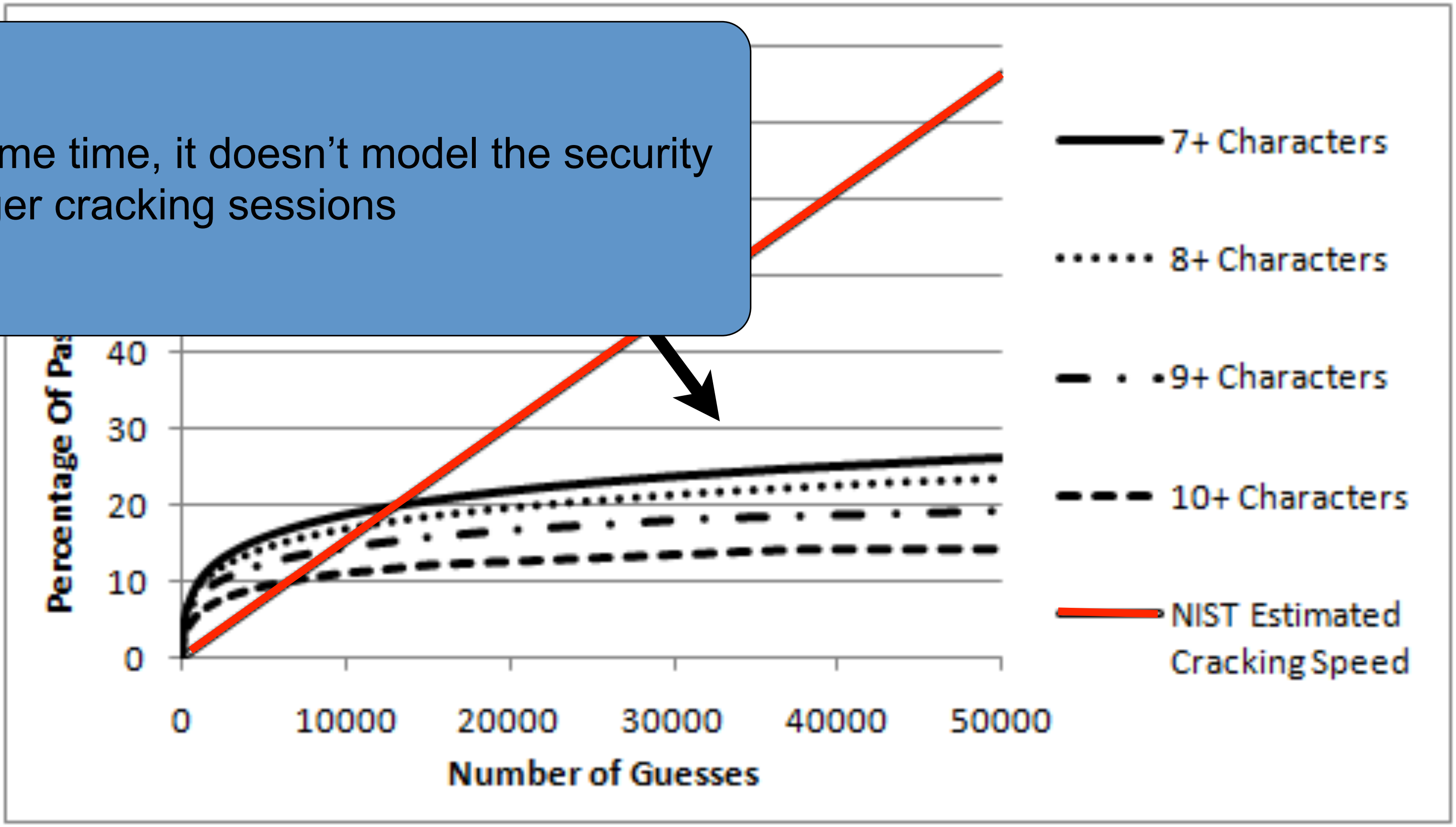


Where this Breaks Down



Where this Breaks Down

At the same time, it doesn't model the security over longer cracking sessions



What this all Means:

Shannon Entropy != Guessing Entropy

Password entropy as defined in NIST 800-63 is not a useful measurement for the defender

Keys vs Passwords

Crypto keys

- Suppose key is 64 bits
- Then 2^{64} keys
- Choose key at random
- Then attacker must try about 2^{63} keys

Passwords

- Suppose passwords are 8 characters, and 256 different characters
- Entropy is $\log_2(b^n)$
- Then $256^8 = 2^{64}$ pwds

  Users do not select passwords at random
- Attacker has far less than 2^{63} pwds to try **(dictionary attack)**

Why not Crypto Keys?

"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations.

(They are also large, expensive to maintain, difficult to manage, and they pollute the environment.

It is astonishing that these devices continue to be manufactured and deployed.

But they are sufficiently pervasive that we must design our protocols around their limitations.)"

Charlie Kaufman, Radia Perlman, Mike Speciner
in "Network Security: Private Communication in a Public World"

How to Store Passwords in the System?

- Store as cleartext
  - If password file compromised, all passwords revealed
- Encipher file
  - Need to have decipherment, encipherment keys in memory
- Store one-way hash of password

Password File

- Bad idea to store passwords in a file
- But need a way to verify passwords
- Cryptographic solution: hash the passwords
  - Store $y = \text{hash}(\text{password})$
  - Can verify entered password by hashing
  - If attacker obtains password file, he does not obtain passwords
  - But attacker with password file can guess x and check whether $y = \text{hash}(x)$
  - If so, attacker has found password!

Dictionary Attack

- Attacker pre-computes $\mathrm{hash}(x)$ for all x in a dictionary of common passwords --- Rainbow Table
- Suppose attacker gets access to password file containing hashed passwords
  - Attacker only needs to compare hashes to his pre-computed dictionary
  - Same attack will work each time
- Can we prevent this attack? Or at least make attacker's job more difficult?

Password File

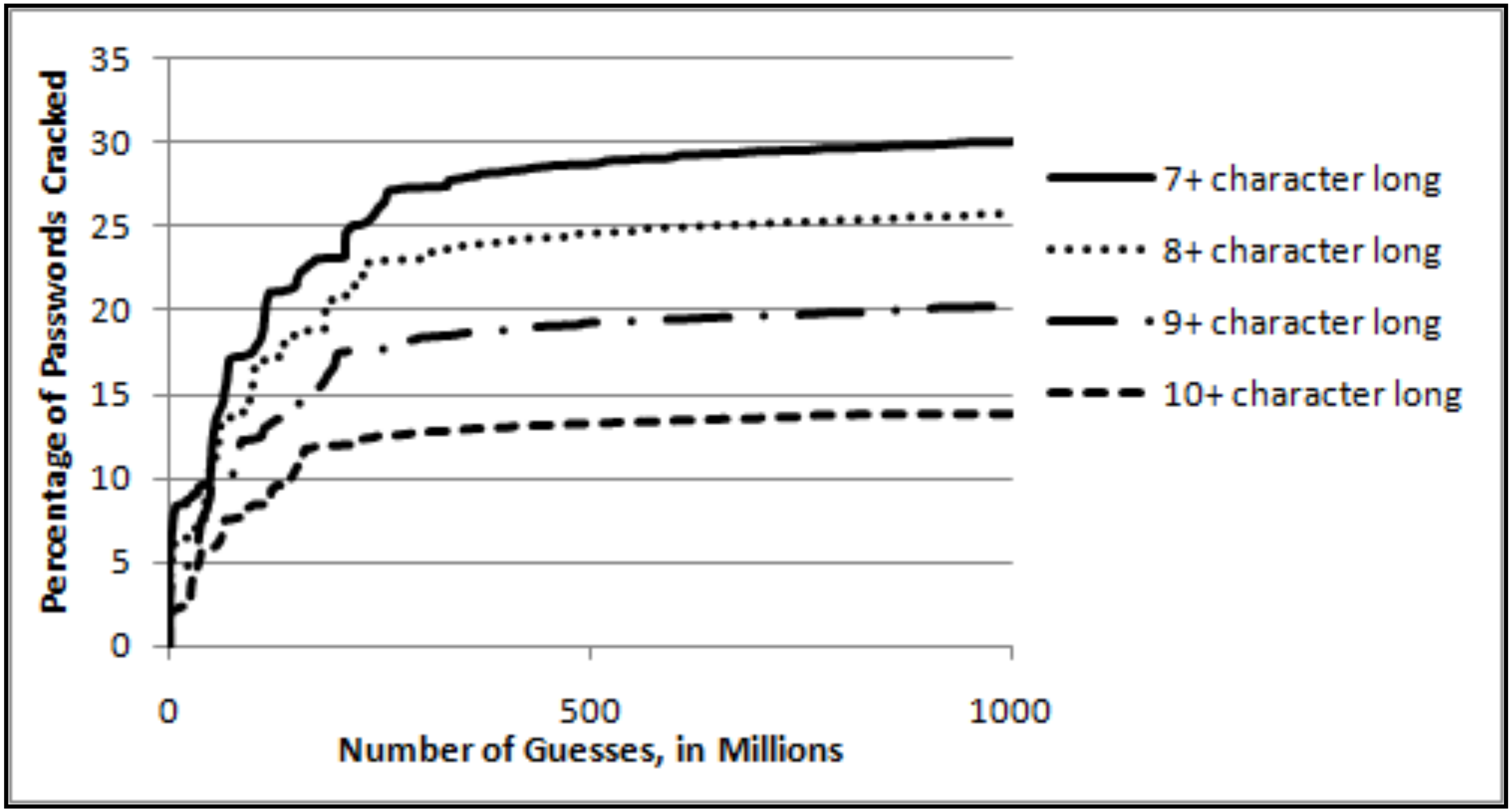
- Store hashed passwords
- Better to hash with salt
- Given password, choose random s, compute

$$y = \text{hash}(\text{password}, s)$$

  and store the pair (s,y) in the password file
- Note: The salt s is not secret
- Easy to verify password
- Attacker must recompute dictionary hashes for each user — lots more work!

UBC ece

Standard Offline Password Cracking Attack



Assumptions for Password Cracking

- Passwords are 8 chars, 128 choices per character
  Then $128^8 = 2^{56}$ possible passwords
- Attacker has dictionary of 2^{20} common pwds
- Probability of 1/4 that a pwd is in dictionary
- Work is measured by number of hashes

Password Cracking

- Finding single password without dictionary
  - Must try $2^{56}/2 = 2^{55}$ on average
  - Just like exhaustive key search
- Finding single password with dictionary
  - Expected work is about

$$1/4\ (2^{19}) + 3/4\ (2^{55}) = 2^{54.6}$$

  - But in practice, try all in dictionary and quit if not found — work is at most 2^{20} and probability of success is $1/4$

password cracking without dictionary

- there is a password file with 2^{10} pwds
- goal: Find any of 1024 passwords in file

Without dictionary:

- **assume all 2^{10} passwords are distinct**
- **need 2^{55} comparisons before expect to find password**
- **if no salt, each hash computation gives 2^{10} comparisons ⇒ the expected work (number of hashes) is**

  $2^{55}/2^{10} = 2^{45}$
- **if salt is used, expected work is 2^{55} since each comparison requires a new hash computation**

password cracking with a dictionary

- Find any of 1024 passwords in file
- With dictionary
  - Probability at least one password is in dictionary is $1 - (3/4)^{1024} = 1$
  - We ignore case where no password is in dictionary
  - If no salt, work is about $2^{19}/2^{10} = 2^{9}$
  - If salt, expected work is less than 2^{22}
  - Note: If no salt, we can precompute all dictionary hashes and amortize the work (Rainbow Tables)

UBC ece

on-line password guessing attacks

features of on-line guessing

- no need to have access to the password database
- limited number of attempts
  - but can be distributed through IP addresses (botnets) or accounts
  - lock out can lead to DOS on the account(s)

32

defence techniques

- making users to choose stronger passwords
- automatic turing test (ATT), e.g., CAPTCHA after so many failed attempts
- account locking
  - DOS is a challenge
- delaying server response
  - ineffective against botnets
- 2-step verification
  1. register a mobile phone on the account
  2. provide password and SMS code received on pre-registered phone
  3. indicate if next time you will be asked for the code to authenticate on this device

users and passwords

over 0.5 M passwords

- The average user has 6.5 passwords,
each of which is shared across 3.9 different sites.
- Each user has about 25 accounts that require passwords, and types an average of 8 passwords per day.
- Users choose passwords with an average bitstrength 40.54 bits.
- The overwhelming majority of users choose passwords that contain lower case letters only
(i.e., no uppercase, digits, or special characters) unless forced to do otherwise.
- 0.4% of users type passwords (on an annualized basis) at verified phishing sites.
- At least 1.5% of Yahoo users forget their passwords each month.

source: Florencio, D. and Herley, C. “**A large-scale study of web password habits**,” In Proceedings of the 16th international Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 657-666. DOI= http://doi.acm.org/10.1145/1242572.1242661

Other Password Issues

- too many passwords to remember
  - Results in password reuse
    - Why is this a problem?
    - compromising important accounts via “junk” ones
- failure to change default passwords
- social engineering
  - phishing
- keyloggers
- resetting/recovering password by guessing backup questions
- error logs may contain “almost” passwords
- bugs, keystroke logging, spyware, etc.

users choose same/weak passwords

RockYou	Faithwriters	MySpace
123456	123456	password1
12345	writer	abc123
123456789	jesus1	fuckyou
password	christ	monkey1
iloveyou	blessed	iloveyou1
princess	john316	myspace1
1234567	jesuschrist	fuckyou1
rockyou	password	number1
12345678	heaven	football1
abc123	faithwriters	nicole1

the most frequent passwords for different sites

Influencing Users' Choices of Passwords

Types of Password Creation Policies

- **Explicit**
  - “Your password must be 8 characters long and contain a digit”
- **External**
  - Part of the password is assigned to you, aka a system generated password or two factor authentication
- **Implicit**
  - “Your password isn’t strong enough, choose another”
  - Example: Blacklists

Explicit Policies

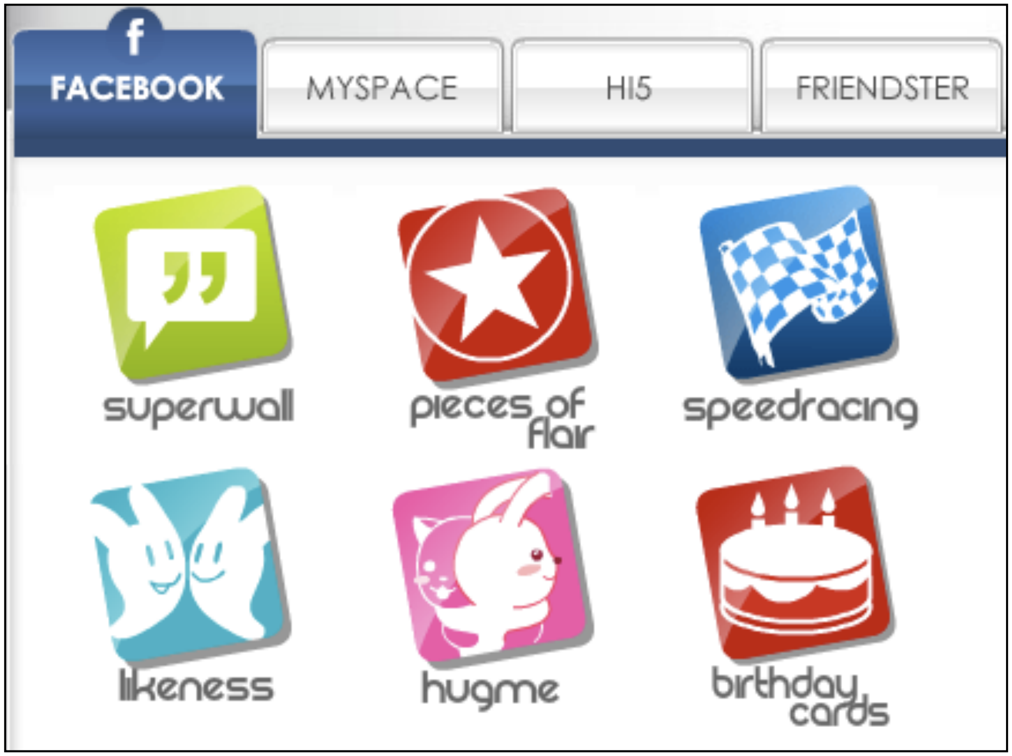
From:
Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords

---

Matt Weir - Florida State University
Sudhir Aggarwal - Florida State University
Michael Collins - Redjack LLC
Henry Stern - Cisco Ironport Systems

Presented at Computer and Communications Security (CCS) Conference, October 2010

The RockYou List



- Provided widgets for most of the major social networking sites
- Hacked in November 2009
- Over 32 million plaintext passwords were released

The PhpBB List

- Development site for the popular phpbBB bulletin board software
- Hacked in January 2009
- Over 259k unsalted MD5 hashed passwords, and another 83k salted passwords

And Many Others:

Singles.org

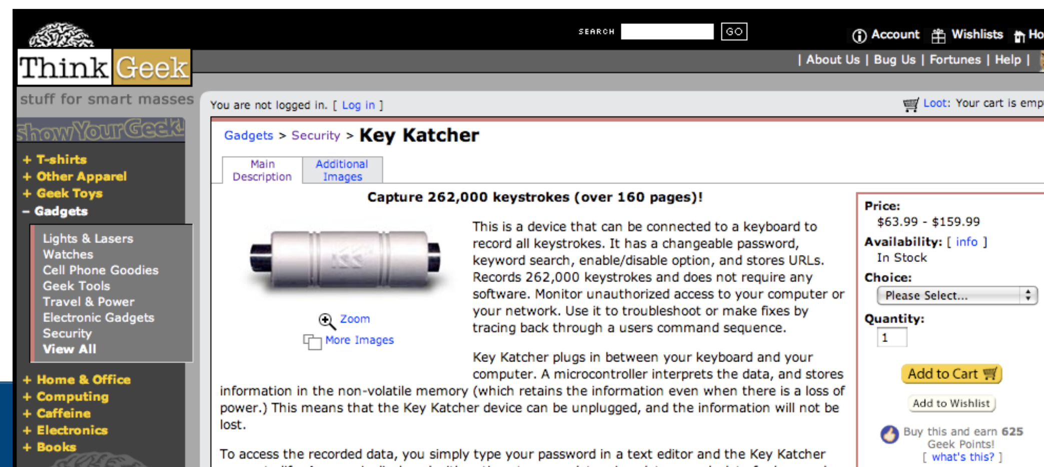
FaithWriters

NeoPets

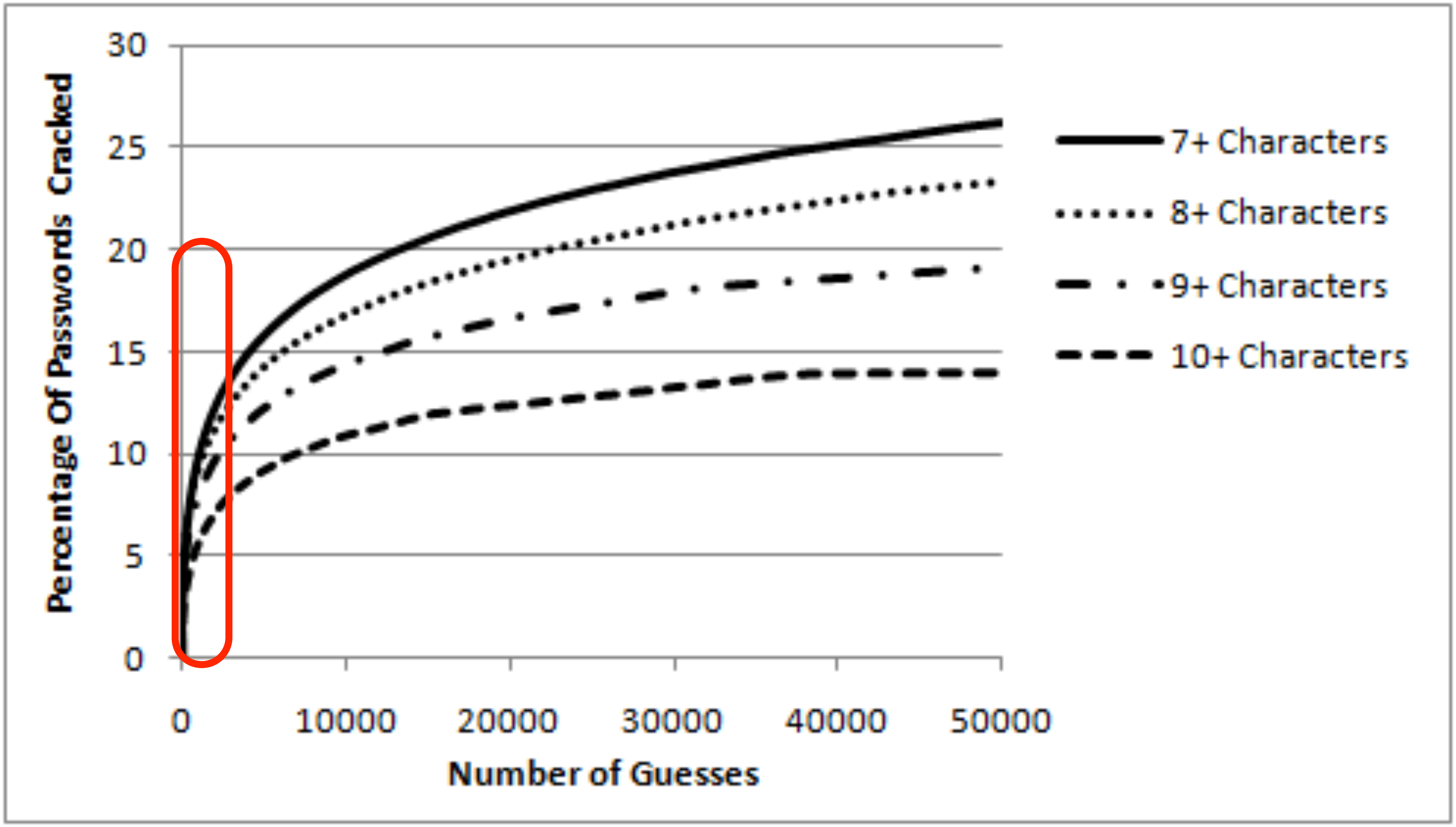
MySpace

Full Disclosure:

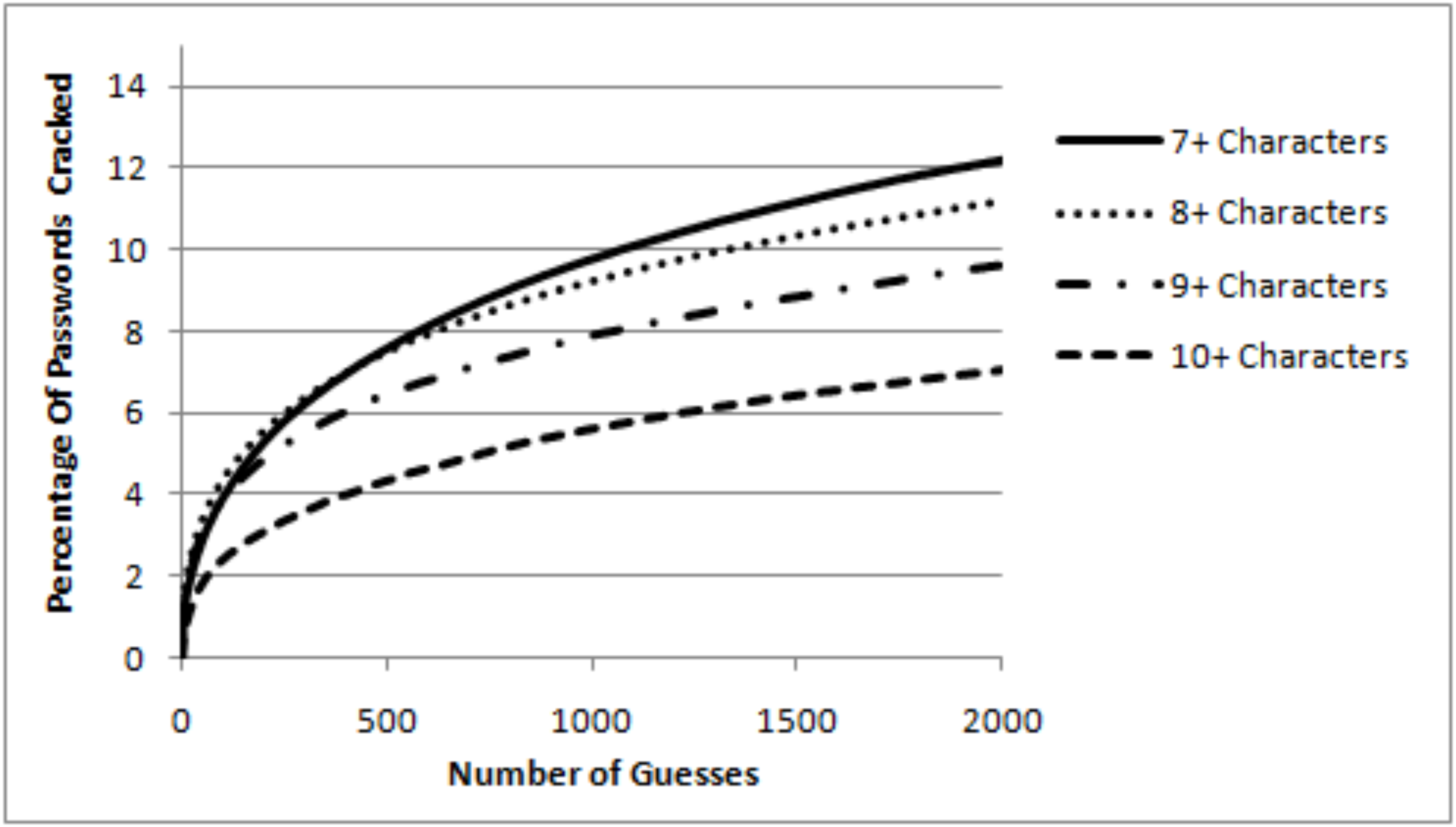
- Password strength rarely matters in an online attack
- More common attacks take advantage of:
  - Password reuse
  - Malware
  - Phishing attacks



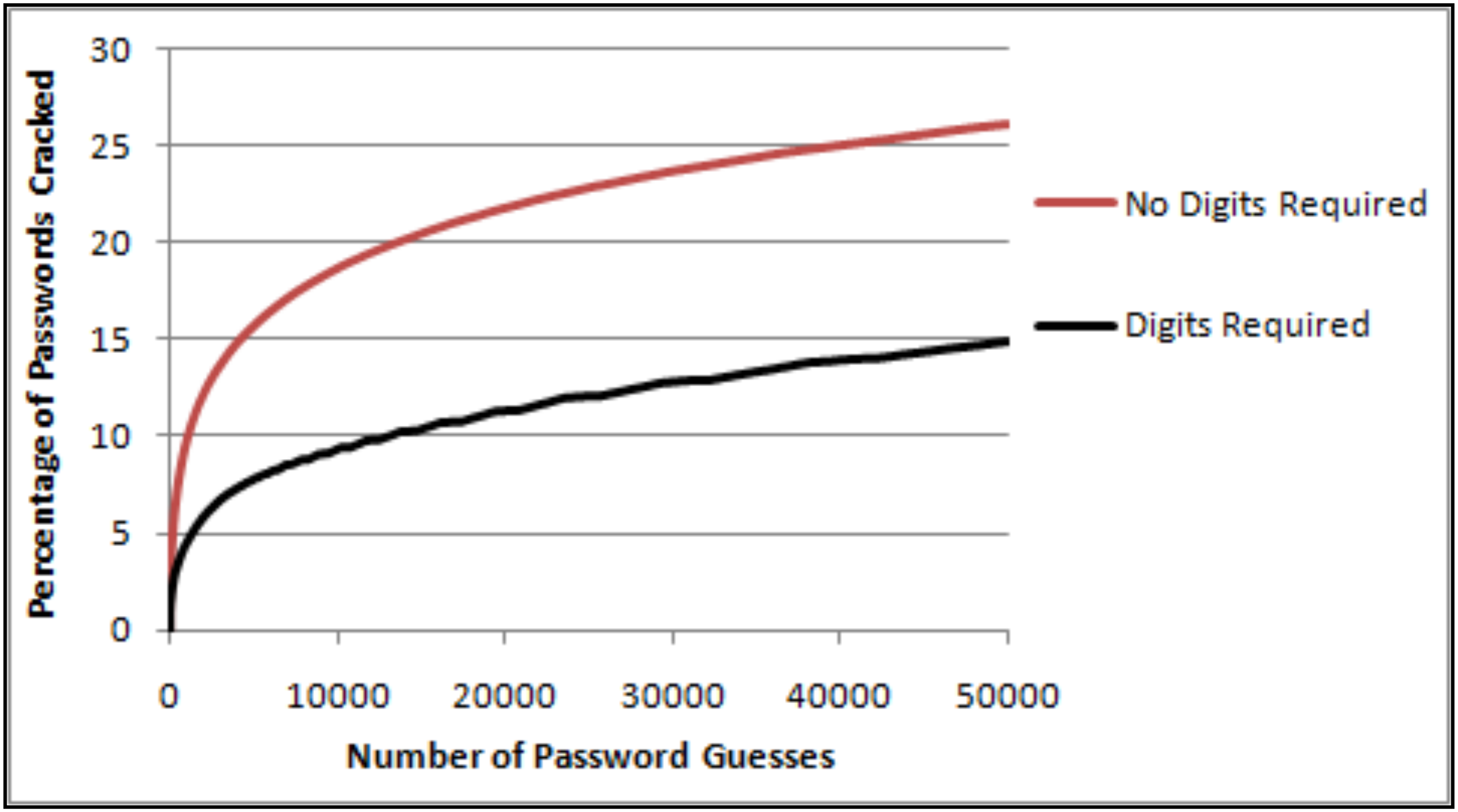
Effect of Password Length



An Even Shorter Cracking Session:



The Effect of Requiring Digits



How Digits were Used:

After	password123	64.28%
All Digits	1234567	20.51%
Other	passw0rd, pass123word, p1a2ssword...	9.24%
Before	123password	5.95%

*Taken from 7+ character long passwords that contained at least one digit

Top 10 Digits From the RockYou Training List

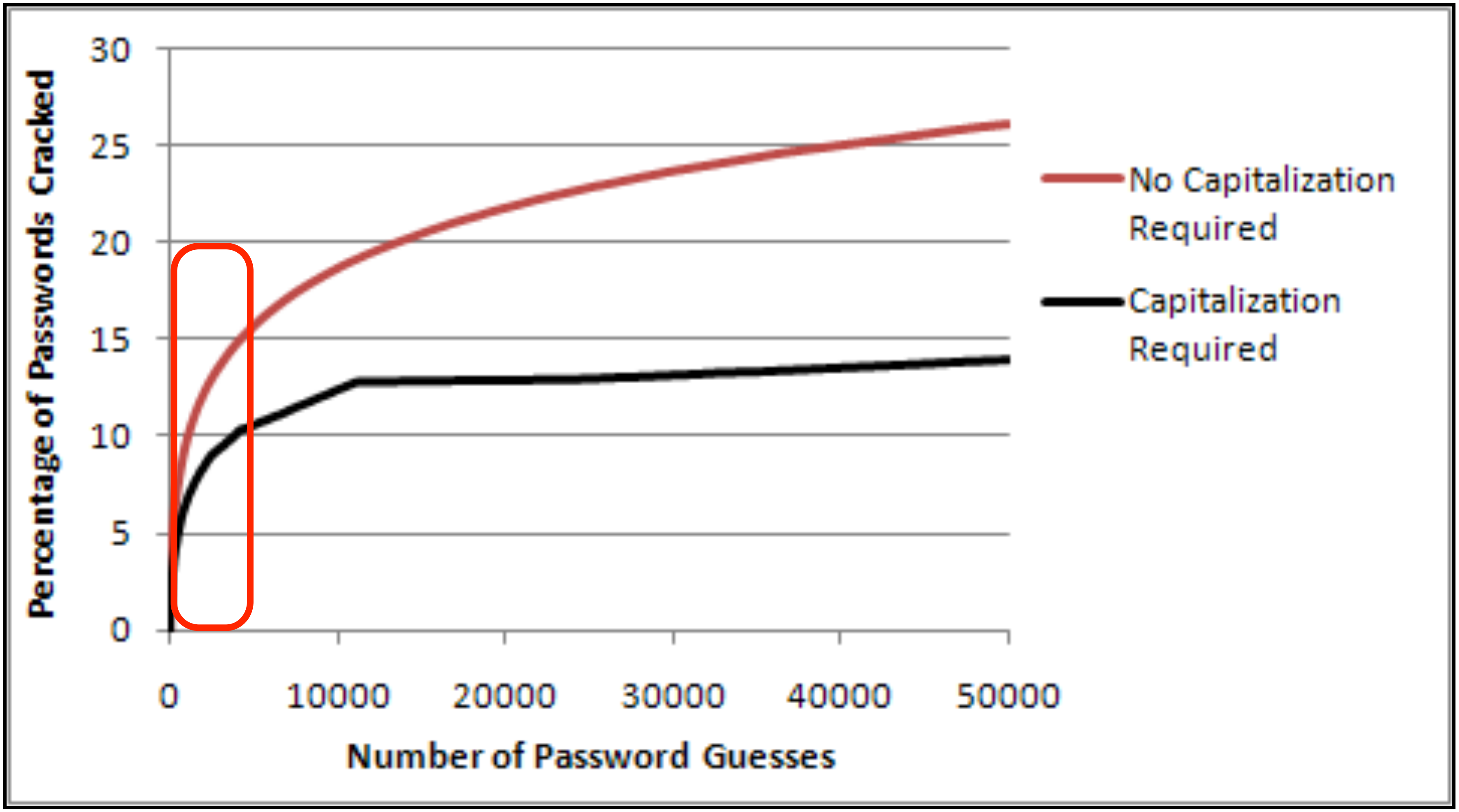
#1	1	10.98%
#2	2	2.79%
#3	123	2.29%
#4	4	2.1%
#5	3	2.02%
#6	123456	1.74%
#7	12	1.49%
#8	7	1.2%
#9	13	1.07%
#10	5	1.04%

26.72% of All Digits

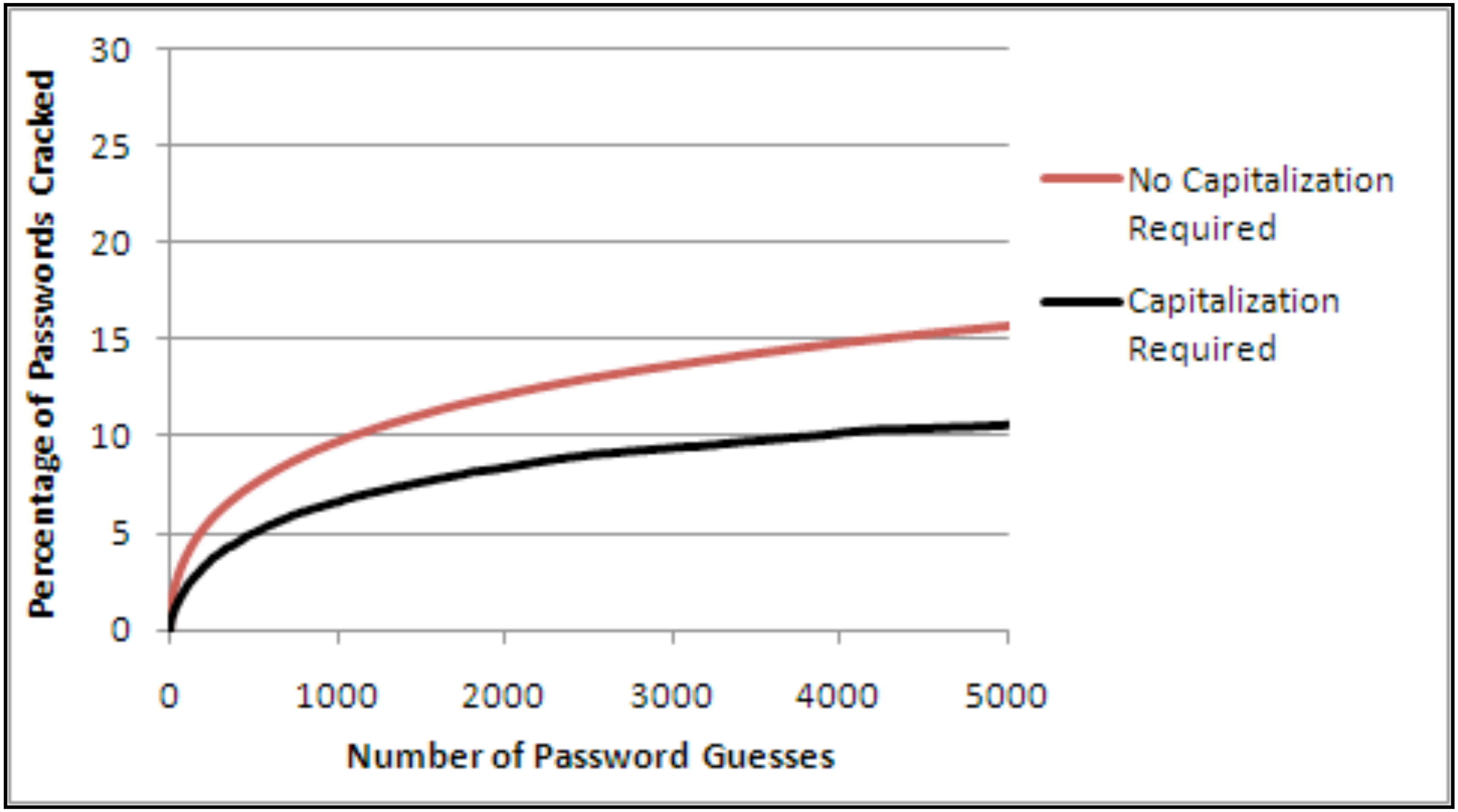
UBC

ece

When Uppercase Characters are Required



Requiring UpperCase - Shorter Cracking Session



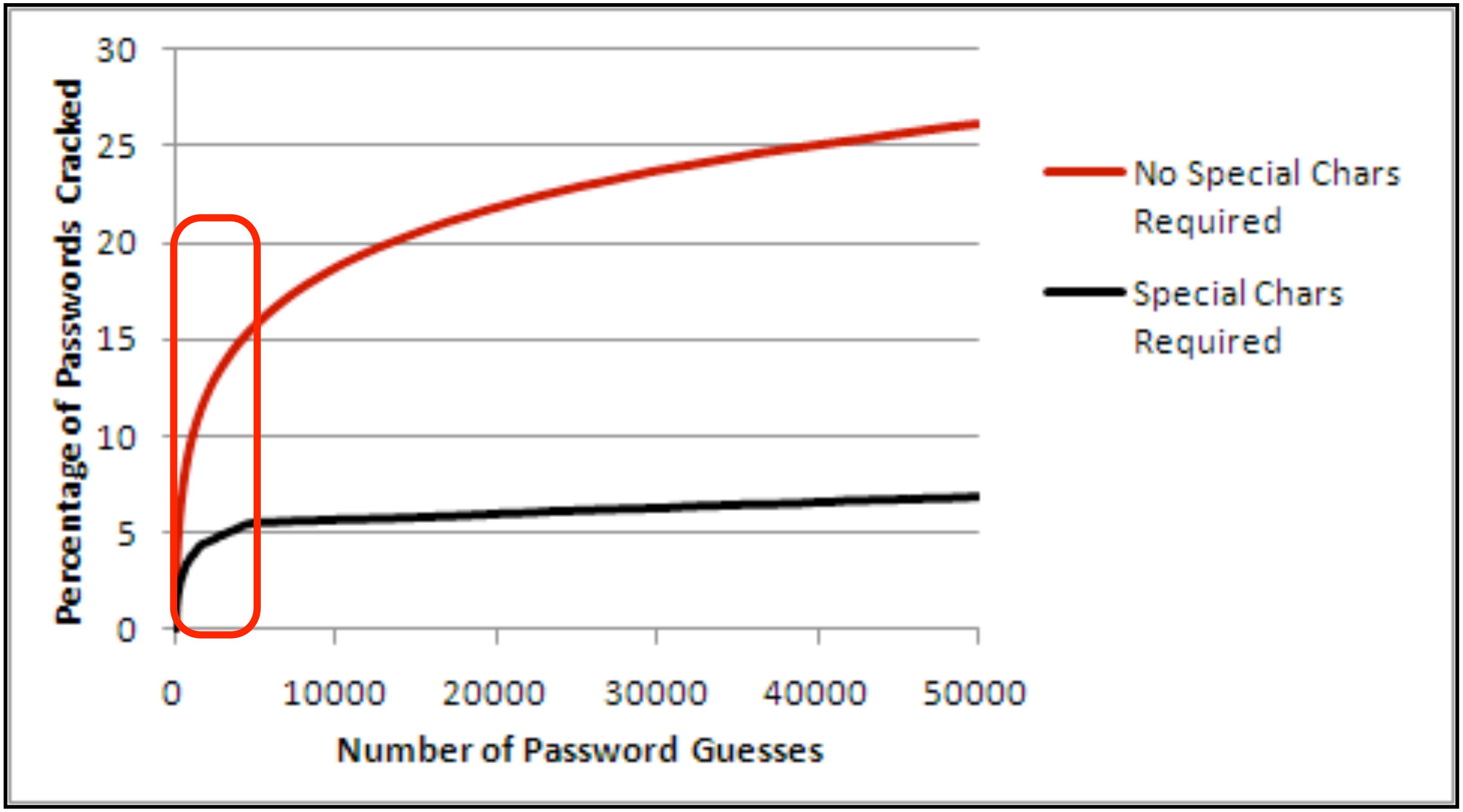
Top Ten Case Mangling Rules of 7 Char Strings

String: U=Upper, L=Lower	Probability
UUUUUUU	53.56%
ULLLLLL	35.69%
ULLLULL	1.05%
LLLLLLL - aka passwor!D	1.03%
ULLLLLU	0.9%
ULLULLL	0.85%
ULULULU	0.68%
LLLLLLU	0.62%
UULLLLL	0.61%
UUULLLL	0.59%

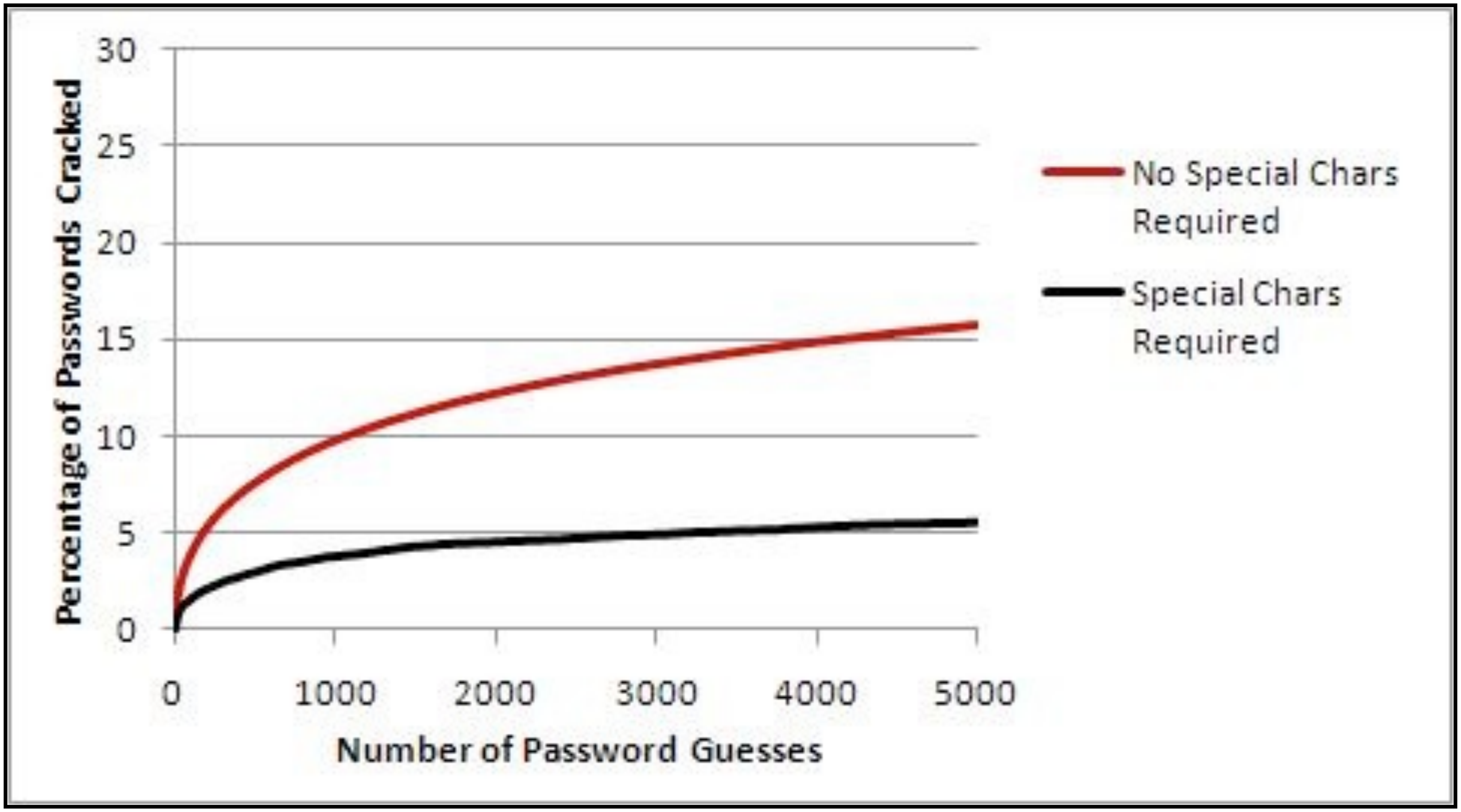
UBC

ece

When Special Characters are Required



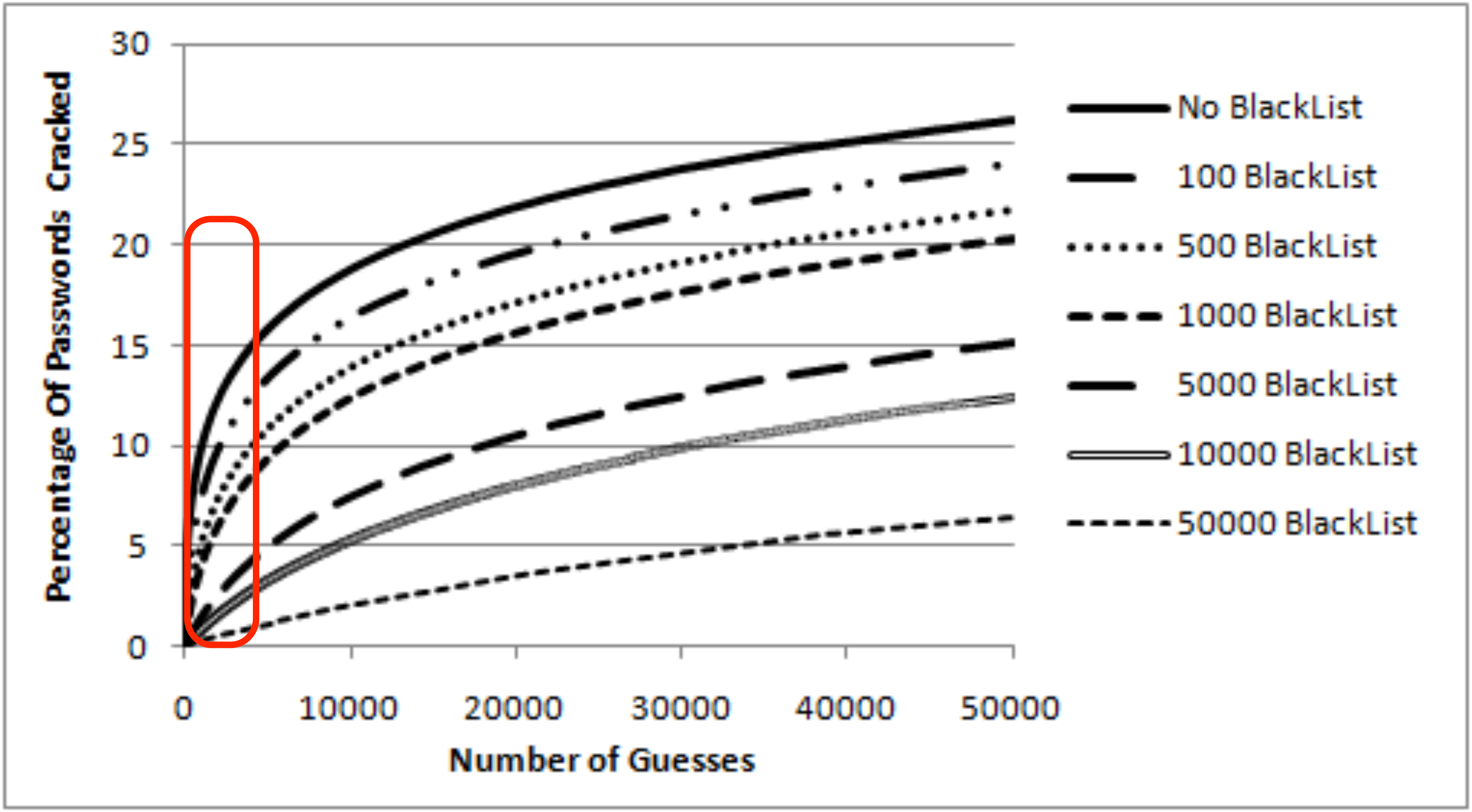
Special Chars Required - Shorter Cracking Session



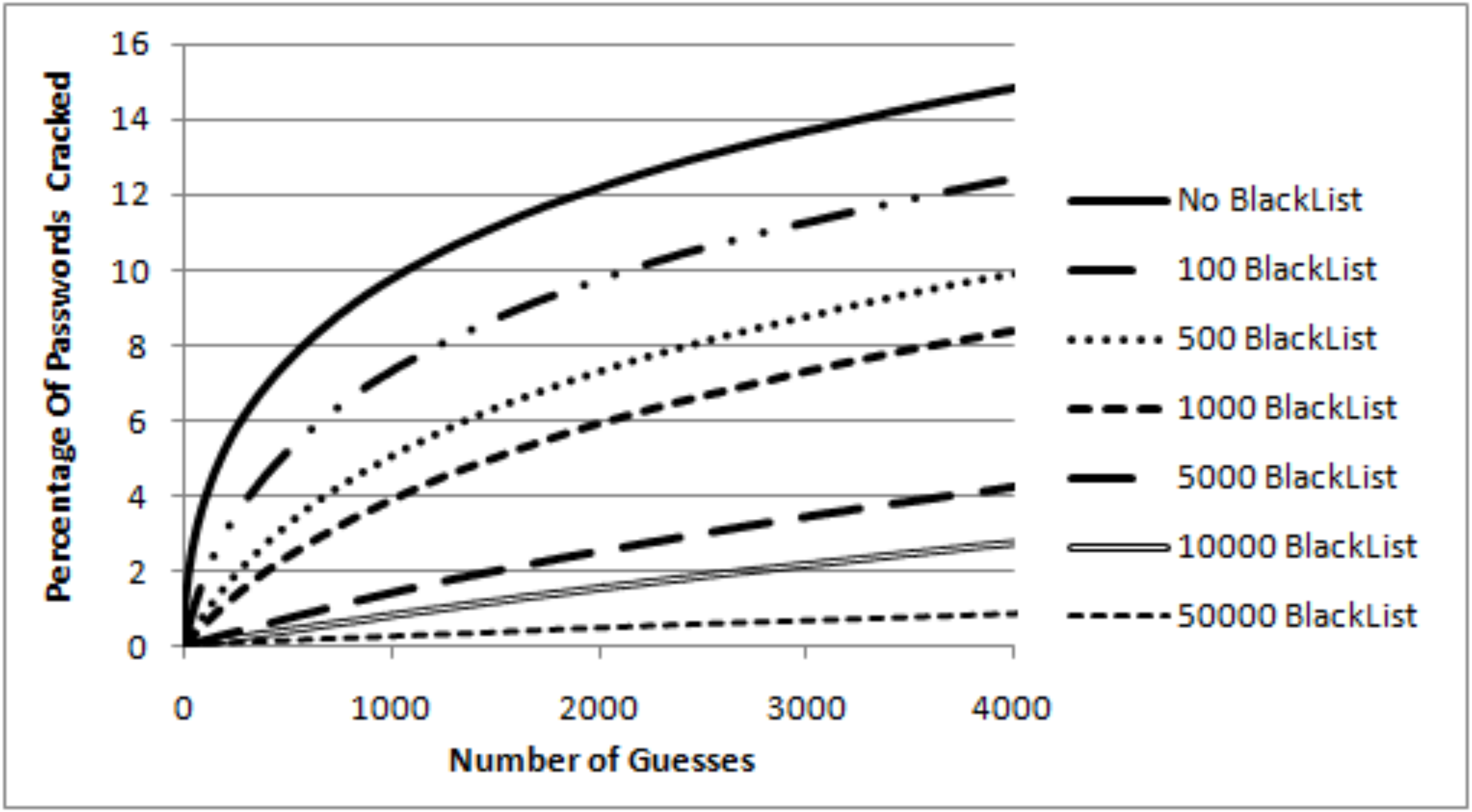
Top Ten Structures for Special Characters

String: A=Alpha, D=Digit,	Probability
AAAAAAS	**28.5%**
AAASAAA	7.87%
AAAASDD	6.32%
AAAAASD	6.18%
AASAAAA	3.43%
AAAASAA	2.76%
AAAAASA	2.64%
SAAAAAS	2.5%
ASAAAAA	2.38%
AAAAASS	2.17%

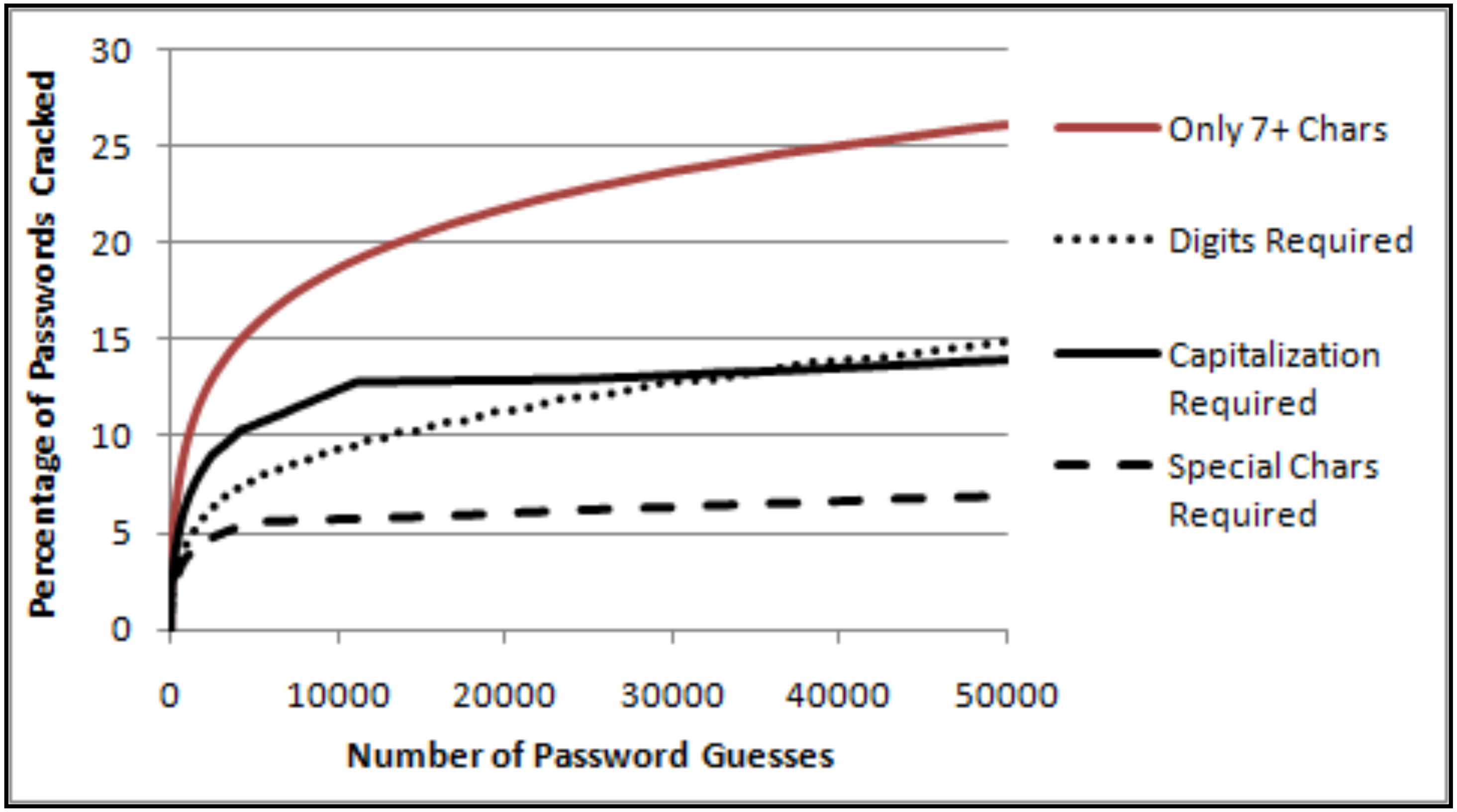
The Effect of BlackLists



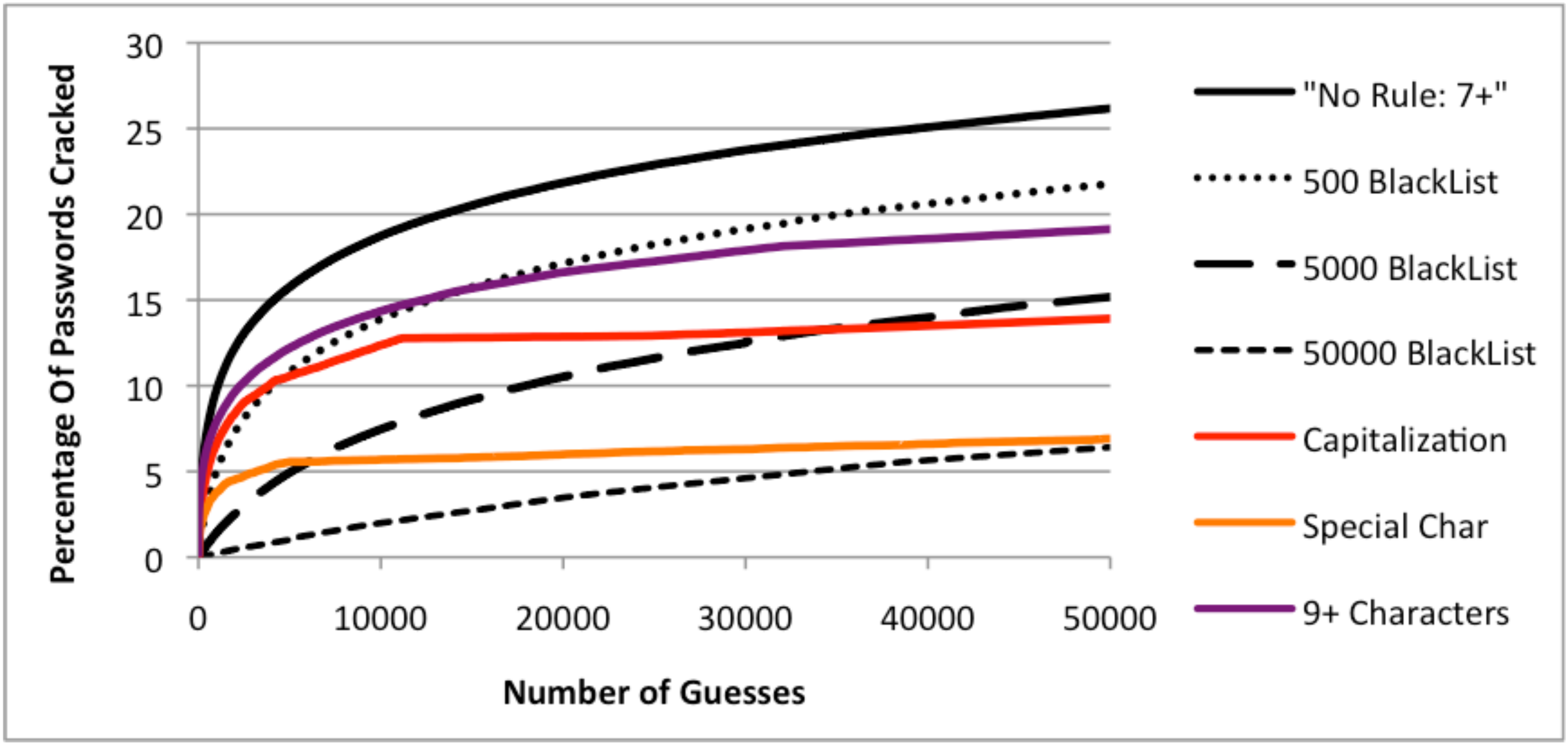
A Closer View:



Comparison of Different Password Requirements

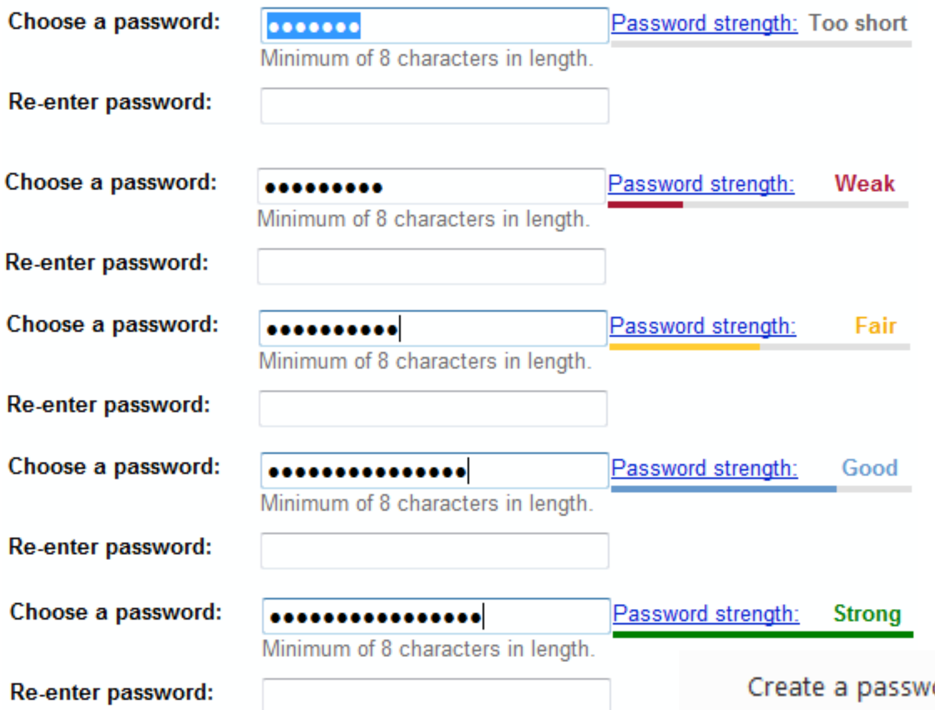


Common Mangling Rules and BlackLists

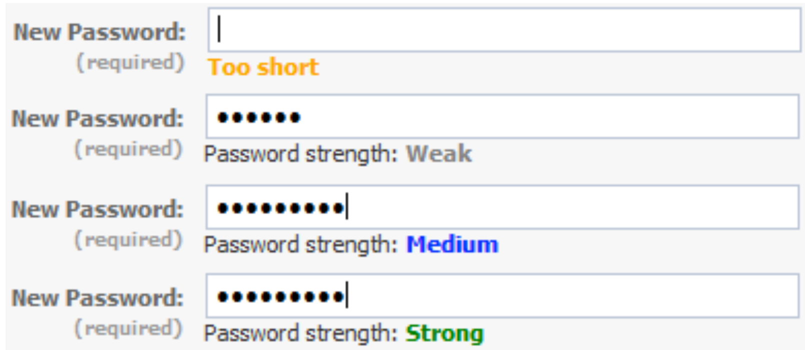


Implicit Policies

Password Strength Meters

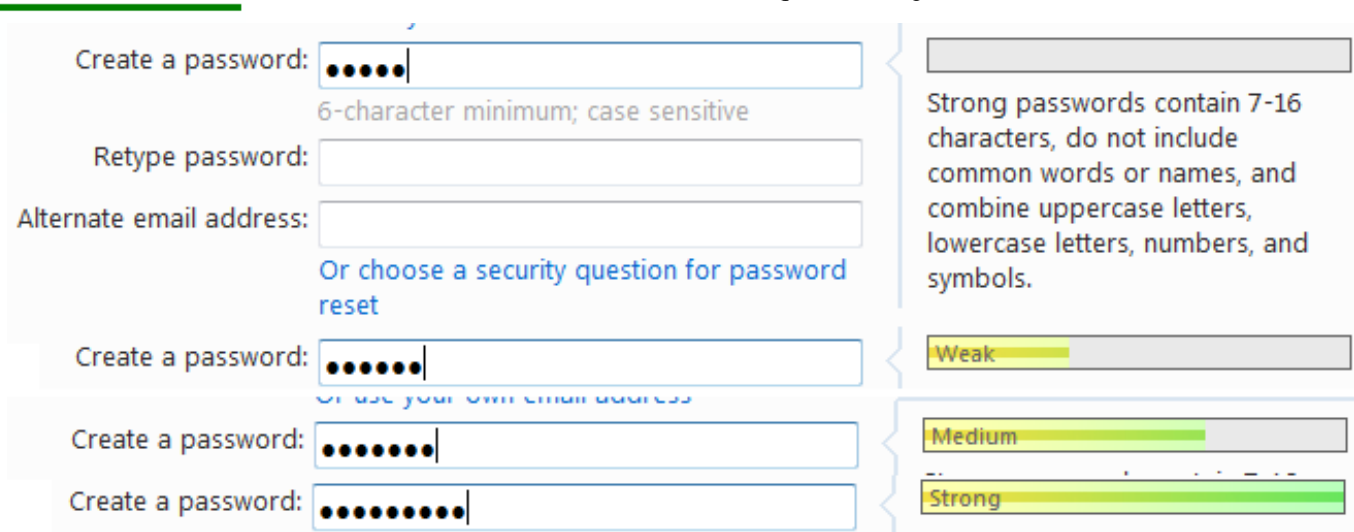


Gmail



Facebook

MSN Live



heuristics of password meters

Password	Ideal	Markov	NIST	MS	Google
`password`	9.09	9.25	21	1	1
`password1`	11.52	11.83	22.5	2	1
`Password1`	16.15	17.08	28.5	3	1
`P4ssw0rd`	22.37	21.67	27	3	1
`naeemha`	21.96	28.42	19.5	1	0
`dkriouh`	N/A	42.64	19.5	1	0
`2GWapWis`	N/A	63.67	21	3	4
`Wp8E&NCc`	N/A	67.15	27	3	4

UBC ece

62

summary

- 3 authentication factors
  - best practice: 2-factor authentication
  - one-time passwords + PINs
- no ideal solution
- passwords are here to stay
  - usability and security issues
- off-line guessing attacks
  - salting + strong passwords
- on-line guessing attacks
  - CAPTCHAs
  - 2-step verification
- password policies
  - explicit, external, implicit
  - password meters
  - blacklisting most popular passwords